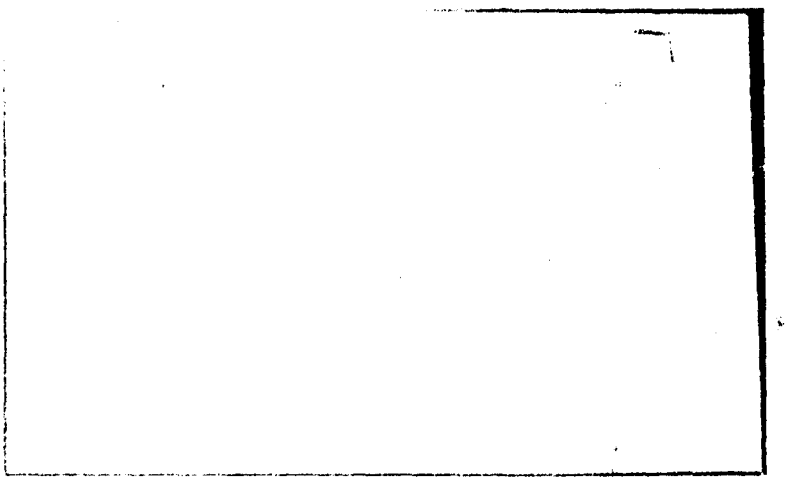
SOME CONSIDERATIONS AND MODELS FOR THE DISTRIBUTION OF A DATA BASE

TEXAS UNIVERSITY AT AUSTIN

MAY 1976



CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Distributed Data Bases						
Data Base Management Systems						
Computer Network & Distributed Systems						
Mathematical Modeling						

Research Report CCS 279

SOME CONSIDERATIONS AND MODELS FOR THE DISTRIBUTION OF A DATA BASE

by

Joyce Elam
Joel Stutz

May 1976

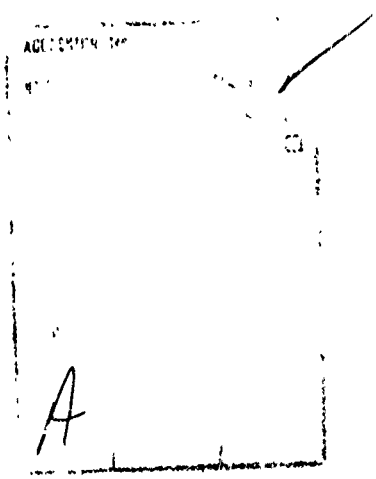
This research was partly supported by Project NR047-021, ONR Contracts N00014-75-C-0616 and N00014-75-0569 with the Center for Cybernetic Studies, The University of Texas. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 20 E
The University of Texas
Austin, Texas 78712
(512) 471-1821

ABSTRACT

This paper presents a survey of current research in the area of distributed systems and discusses some possible areas of future research. The trend toward data base management systems which are responsible for the creation, accessing, and maintenance of a large collection of information containing complex interrelationships is prevalent in centralized systems today. It seems that the same advantages which led to the growth of data base management systems in centralized systems can be applied in distributed systems as well. Most current research (particularly in the area of modeling) views a distributed system in terms of independent data files We address some of the problems involved when the distributed system is viewed in terms of an integrated data base.



SOME CONSIDERATIONS AND MODELS IN THE DISTRIBUTION OF A DATA BASE

Introduction

Early computerized information systems were typically implemented as a centralized batch processing computer installation with the data files and applications programs all resident at the central site. Remote users were denied direct access to the data processing facility. Source documents were sent either by mail or truck to be keypunched and processed. Reports were returned in the same manner. Obviously this method of getting information was not timely.

The first step in providing some form of direct access to a computer facility for these users was the development of computer-communication networks employing terminal devices (CRT's, teletypes) at the remote sites. All processing power continued to be concentrated at the central site. This type of network where all communication lines flow into and out of a central location is often referred to as a "star" or "hub and spoke" configuration.

Due to advances in hardware technology, it became economically feasible to replace terminal devices with independent minicomputers and intelligent terminals. A network consisting of two or more independent computers (nodes) connected by one or more communication channels will be referred to as a "computer network." In this context, the term computer is meant to include intelligent

terminals, minicomputers, and microcomputers as well as the larger processors. The ability for local as well as central site data storage and processing, therefore, exists in a computer network.

In a computer network, the topology can be separated into two broad categories--centralized or decentralized. In a centralized computer network, the basic star pattern of the terminal network is used in which subordinate computers are all connected to a main common computer. A disadvantage of this topology is the lack of any direct communication between subordinate nodes. The decentralized configuration eliminates the central or main computer and directly connects computers of more or less equal rank. In a decentralized computer network the capability exists for not only local data storage and processing but also for the direct sharing of resources among all the nodes.

Organizations have computer system resources of data, processes, and hardware. Computer networks permit similar but independent organizations (such as those in the ARPA network) to share their resources or permit organizations to partition and distribute their resources, normally held at a central location, among the nodes in the network. Organizations may distribute their resources among regional locations which are geographically scattered. This would enable the data and processes to be placed at or near locations which use them most frequently. Organizations may also distribute their resources on a computer network which resides in one location. This is particularly applicable to large data bases where query response time can be shortened

through the use of parallel search techniques.

Irrespective of the type of network employed, advantages in the areas of security, flexibility, and reliability can be gained through the distribution of resources [11]. In the area of security, the semiautonomous nature of the nodes and the controlled communication paths provide numerous avenues for effective system security. Distributed systems offer advantages for both hardware and software flexibility. Software flexibility includes the capability for local source data processing as well as providing access to specialized software at other nodes. Hardware flexibility permits access to unique hardware configurations as well as providing for smooth transitions to state-of-the-art technology since individual subsystems may be replaced with negligible effects on the other subsystems. Given a certain amount of homogeneity among the nodes of a network, it is feasible for one node or subsystem to assume the task of another in the event of failure, thus increasing the level of reliability.

The design problems in a computer network composed of independent organizations are mainly concerned with the topology and communication feasibility (e.g. protocols, format compatibilities, etc.) of the network; the resources to be shared are assumed to pre-exist at the nodes in the network. This is not the case when designing a distributed system for a single organization. The resources once stored at a centralized site must be partitioned and distributed throughout the network. Since there are many possible ways to partition and distribute the resources, the

major consideration is how to optimally allocate the resources and provide efficient access methods. The purpose of this paper is to discuss some of the considerations involved in designing a distributed system, to survey the literature pertaining to resource allocation, and to outline some further areas for research.

The information in an organization prior to distribution can generally be viewed as existing in two different formats. It may be in the form of independent files which were most likely created and organized for specific applications, or the information may be organized as an integrated database.[1] In either case, the first step toward distribution concerns the problem of how to partition the data base[2] into physically distributable segments which, though possibly logically associated with each other (as in the case of a database), can be stored on physically separate and independently addressable storage.

Before further discussion on the options for partitioning of a data base, we need to define and briefly discuss the components of data base systems. An important concept in data management systems is the idea that data is either implicity or explicitly described by some type of information structure. The information structure could include data descriptors which describe data elements and the relationships among the elements (i.e., schema),

---

[1]The term database is used to refer to an integrated collection of all the data pertaining to multiple applications which are organized in a manner which reduces redundancy, provides data independence, and only permits common and controlled access to the data.

[2]Data base is defined in a collective sense to include both the independent and integrated information organizations.

descriptors to content (e.g., inverted file), and descriptors to locations (e.g., indexes, directories, etc.). Therefore, the considerations in partitioning and distributing a data base involve the "data," the information structure, and the relationships between them.

One possible solution is to partition the data base by function or process. The rationale in partitioning by function is to include in the same segment data instances that are used by a particular process such as inventory, billing, etc. Independent files are normally segmented by function a priori. Thus the partitioning options in this case concern the possible separation of the data and data descriptors and the possible segmentation of the logical data files into discrete physical segments. In an integrated database, functional partitioning corresponds to the creation of segments containing only the instances of a given record type, and segments containing the corresponding data descriptors.

Another possibility is to partition the data base by characteristic. An example of this could be a credit card data base in which data and data descriptors are partitioned into segments according to the state in which the credit holders reside. This is segmentation across record types such that a given segment contains record occurrences from multiple record types.

We have in no way attempted to exhaust all possible options for partitioning of a data base (this being an area for further

research), but have merely included some possible options for illustrative purposes. The main point to be made is that regardless of the option chosen the partitioning results in a set of independently distributable segments.

Given a set of distributable segments, the next major consideration in the design of a distributed data base is that of distributing the segments in order to obtain a "good" or optimal network according to some criteria. As with partitioning, the method of distribution may be by function or by characteristic. Within these broad categories there are a number of further considerations.

One concerns the proximity and degree of dependency of data and associated data descriptors to the processes which use them. In this regard, the additional issue of redundancy arises since increased redundancy will enhance the answering of queries but will confound the performance of updates. Regardless, there will undoubtedly arise situations in which the process and its required data are physically separate (different nodes). In order for nodes to access information stored elsewhere in the network, some location method must be provided. In addition to location descriptors which existed prior to distribution of the data base, the act of distribution imposes the need for locators which identify the nodes at which the segments containing the required information reside. These locators may be classified according to the type of directories maintained in the system: full directory, partial directory, or no directory [9].

A full directory is a directory that lists every segment in the network and indicates where they are physically located. This allows a segment to be located by accessing at most one directory and allows the nonexistence of a segment to be detected immediately. However, the problem of reliability arises when only one copy of the full directory is kept and the problem of consistency exists when duplicate copies are maintained.

A partial directory is a directory that lists every segment resident at a particular node or group of nodes. For instance, each node could have a partial directory for its segments, or a node in each region could have a partial directory for all the segments located in that region. There are many further possibilities for hierarchies of partial directories. A request for a segment would first result in searching the local partial directory and if necessary the other partial directories. The inter-directory query could be accomplished by a broadcast search, a request sent to all the appropriate partial directories. Another method would be to link the partial directories together in a ring structure and query each one in turn until a positive response is received or a complete circuit of the directories has been made.

No directory implies that the name or some characteristic of a segment identifies its physical location in the network. No processing of directories is necessary to locate this segment but any movement of this segment will result in a change in name which must be known by all users, or the maintenance of some

type of pointer structure to trace the segment's movement.

A particular distributed system is not restricted to using a single directory type. Most likely, a combination of directory types will be utilized in order to permit flexibility in the design of the data base system. The directories may be viewed as distributable segments whose distribution involve the same considerations with respect to redundancies, dependencies, etc. as any other segment. Ideally, the end user should not be concerned with the physical whereabouts of the records or directories to the records he wishes to access.

Given that a certain process or query requires access to segments (processes, data, data descriptors) which may reside on separate nodes, decisions must be made concerning the manner in which the segments are brought together. These decisions are dependent upon the degree of separation within the network and have important implications on the level of simplification or complexity of the access paths. A wide range of variation in access paths can exist depending on the location of the processing program, the data, the data descriptors, and the amount of data to be transfered. When a program requests data, a decision must be made on when to send the data and how much of the data to send. This arises due to the fact that processes do not necessarily request data solely when the need arises and often times use none, part of, or some form of summarization of the data requested. One possibility would be to send all data requested upon initial request if there is a greater than

zero probability that the program will use the data. In this case, the processing program, data descriptors, and data would be joined together at a node for the duration of the execution of the program. This type of access should only be efficient in the case of a program utilizing sequential access to large amounts of data. This would involve the transfer of large amounts of data but would result in access simplification during the execution of the program. Another alternative would be to send only the record occurrences explicitly required by the processing program. In this case, the processing program, data descriptors, and data might reside on separate nodes with an access request to the data descriptors and an access request to the data being made each time the processing program requests some data from the network. This would involve the transfer of smaller quantities of data but would increase the access complexity needed to retrieve data. The degree to which a system can reduce the quantity of data transfered with respect to the quantity of data examined is sometimes referred to as the data reduction ratio. Thus a query for the total amount charged by all credit card holders in Texas might be handled in several ways. One extreme would involve the transfer of all record occurrences containing charges to the processing program where they are then totaled; the other extreme would involve the summation at the data resident nodes and the transfer of sub-totals. Obviously, there is a great deal of difference in the data reduction ratio achieved.

In addition to the preceding considerations in the design of a distributed data base system, there are a host of additional factors which must be taken into account. The decisions necessarily involve numerous tradeoffs due to the competing nature of the design parameters. Therefore one must assign relative priorities in addition to possible minimum requirements for all parameters under consideration. This suggests the use of some tool for choosing a "good" or optimal configuration from among the set of feasible alternatives. Assuming that priorities and requirements can be quantified, a mathematical model is a natural vehicle for expressing the relationships and mathematical programming a natural technique for obtaining a solution. Thus far the research in this area has focused on the file(segment) allocation problems involved in a distributed data base system. The next section surveys the research to date.

Survey of Models

The file allocation problem was first investigated by W. Chu [6,7]. Given a number of computers in a network which process common files, Chu developed a mathematical model for determining an allocation of these files which minimizes the overall operating cost of the network while satisfying certain requirements. His model determines the allocation of multiple copies of multiple files in a network where the topology is assumed fixed, and each pair of computers has direct communication paths. In determining the overall operating cost of the network, the model considers storage cost, transmission cost, file length,

number of requests for queries and updates to the files, and the number of copies of each file stored. The constraints of the model are (1) the amount of storage needed at each computer does not exceed the available storage capacity, (2) the expected time to access each file is less than a given bound, and (3) the availability (the portion of the time the system is operating) of each file is greater than a given level. The formal mathematical model is given in Figure 1.

The model formulated by Chu is a nonlinear zero-one programming model. The first step in his analysis deals with the determination of values for the unknown input parameters used in the model. Queueing delay equations using Poisson assumptions were developed for determining the expected time for the ith computer to retrieve the jth file from the kth computer (a_{ijk} in the model). Next, file availability equations were developed for expressing availability of a file in terms of the reliability of the computers and communication channels, average repair time, and the number of copies of each file. Given a desired level of availability for each file and a measure of the reliability of the system, the required number of copies of each file can be determined. The availability constraint, therefore, is not used explicitly in the model, but is used prior to model solution to determine the requisite number of copies of each file. Next, in order to make the problem amenable to solution, the nonlinear zero-one programming problem is reduced to a linear zero-one programming problem.

Objective:

$$\text{Min} \quad \sum_{i,j,k} c'_{ik} l_j u_{ij} x_{kj} \rho_{ij} + \sum_{i,j,k} \frac{1}{r_j} c'_{ik} l_j u_{ij} x_{kj} (1 - x_{ij}) + \sum_{ij} c_{ij} b_j x_{ij}$$

Subject to:

$$\sum_j x_{ij} b_j \le b_i \qquad 1 \le i \le n$$

$$(1 - x_{ij}) x_{kj} a_{ijk} \le T_{ij} \qquad \forall i \; i \ne k \quad 1 \le j \le m$$

$$\sum_i x_{ij} = r_j \qquad 1 \le j \le m$$

where r_j is determined by

$$A_j = a_p [1 - (1 - a_c a_p)^{r_j}]$$

and

a_{ijk}	expected time for ith computer to retrieve jth file from the kth computer
T_{ij}	maximum allowable retrieval time of the jth file to the ith computer
a_c	availability of computer channel
a_p	availability of requesting computer
A_j	availability level of file j
b_i	available memory size of the ith computer
c'_{ik}	transmission cost per unit length from i to k
c_{ik}	storage cost of unit length of file j at node i
l_j	length of transaction to file j
b_j	length of file j
u_{ij}	transactions from node i to file j
ρ_{ij}	frequency of updates of file j from node i
x_{ij}	= 1 if file j is stored at node i = 0 otherwise
n	number of nodes in the network
m	number of files to be allocated

Figure 1 - Chu's model

An example of a file allocation problem is presented using a three node network and five files. The problem was solved using the Gomory cutting plane technique. This solution method, however, can only be applied to very small problems. Although the model has several limitations, it was the pioneering work in applying mathematical programming to the distribution of data bases.

Based upon the work Chu, Casey in two separate papers developed further models of file allocation in a computer network. In his first paper, Casey [3] addressed the problem of allocating multiple copies of a single file in a computer network of fixed topology. Within this network every node is able to communicate with every other node over direct communication links or through intermediate nodes. As in Chu's model, the objective is to minimize the storage and communication costs of file allocation. There are, however, several differences between the two models. In order to remedy a limitation inherent in Chu's model, Casey includes the determination of the number of redundant copies as a variable in the model. Casey also makes a distinction between query and update activity in the network; in fact, it is the relative tradeoffs between query and update costs based on their respective volumes which determine the number of copies of the file. When making a query, a user is assumed to access only the copy of the file which minimizes his communication cost. Updates require every copy of the file to be accessed. As the number of copies of the file increases, querying costs go down

but update and storage costs go up. In the limiting cases, if no updates are done, minimum cost can be achieved by storing a single copy of the file at some optimal location. Given a certain proportion of update to query transactions, the problem becomes one of determining the number of copies and an associated allocation which minimizes the total cost. The formal mathematical model is given in Figure 2.

Objective:

$$C(I) = \sum_j \lambda_j \min_{k \in I} d_{jk} + \sum_j \sum_{k \in I} \psi_j d'_{jk} + \sum_{k \in I} \sigma_k$$

Choose assignment I such as to minimize C(I)

λ_j = query traffic from node j

d_{jk} = transmission cost from node j to node k for queries

d'_{jk} = transmission cost from node j to node k for updates

σ_k = storage cost at node k

ψ_j = update traffic from node j

Figure 2 - Casey's model

Generally, the methods for obtaining an optimal solution to integer programming problems are both inefficient and costly. Heuristics are normally developed which produce a "good" though nonoptimal solution. By exploiting the special properties of the cost function, Casey developed an efficient search procedure using a form of implicit enumeration which guarantees an optimal solution. If this search procedure becomes too expensive to apply due to the size of the problem, heuristic modifications can be employed to produce a "good" solution.

It was first shown that the proportion of update traffic to query traffic determines an upper bound on the number of the copies present in the least cost network. It was also shown that beginning with a single copy, the cost function monitonically decreases as additional copies of the file are added up to a certain limit after which the cost function monitonically increases. These properties permit the search procedure to examine only a small subset of the 2^n possible assignments of a single file in a n-node network.

The search procedure finds a set of local optima in the following manner:

(1) assign a copy of the file to a particular node and determine cost

(2) consider adding a copy of the file to one additional node and compute the cost

(3) if the cost decreases with the addition of a copy, go to step (2)

(4) if not, the preceding assignment was a local optimum

(5) repeat the above steps until all local optima are found

The global optimum is the minimum of the local optima.

Casey tested his algorithm using five different query/update proportions on a 19-node subset of the ARPA netwrok. In addition, a heuristic modification to the algorithm was applied to the same network utilizing a modified objective function which assumed more economical update patterns between nodes. The heuristic procedure took significantly longer but resulted in substantially less cost.

In his second paper, Casey [4] extended the file allocation

problem to consider how the data base of an organization should be distributed in a computer network, and what type of communications configuration should be implemented in order to satisy users' needs at a minimal cost. The data base is assumed to be partitioned into files that can be stored independently in the network. It is further assumed that the network communications are of the packet-switched variety in which the channel costs are dependent upon capacity. The channel capacity is assumed to be available from common carriers in discrete increments. In constrast to his first model, the costs for queries and updates are not explicitly included in the cost function. Instead, these costs are indirectly reflected in the cost of leased lines since the average transaction volume determines the channel capacity required. Also included in the cost function is a storage cost as in his previous model.

The model is nonlinear and contains both integer and continuous variables. The design variables to be optimized include (1) assignment of each file to a set of locations, (2) selection of network topology, (3) designation of routing paths between communicating nodes, and (4) specification of channel capacity for each link of the network. The model includes two constraint categories. One set of constraints guarantees the delievery and conservation of messages. The other set insures that channel capacity will not be exceeded. The formal mathematical model for this problem is given in Figure 3.

The number of possible networks is immense even for small

Objective:

Min $\sum_{a,k} y_{ak} S_{ak} + \sum_{i,j} d_{ij}(x_{ij})$

Subject to:

(1) no messages lost or gained at an intermediate node

$\sum_j q_{ij}(a,w,k) = 0$ each i,a,w,k , $i \neq w, i \neq k$

$\sum_j u_{ij}(a,w,k) = 0$ "

(2) total query traffic from wth user to all copies of file a

$\sum_k q_{wk}(a,w,k) y_{ak} = Q_{wa}$ each w,a

(3) total update traffic from wth user to each copy of ath file

$\sum_i u_{ik}(a,w,k) = U_{wa} y_{ak}$ each a,w,k

(4) absorption of query traffic at files nodes

$\sum_i q_{ik}(a,w,k) y_{ak} = Q_{wa}$

(5) all updates transmitted to each file copy

$\sum_i u_{ik}(a,w,k) = U_{wa} y_{ak}$

(6) capacity constraint: no overloaded links

$\sum_{a,w,k} [|q_{ij}(a,w,k)| + |u_{ij}(a,w,k)|] \leq C_{x_{ij}}$ each i,j

Given:

k processor nodes , $k = 1, 2, \ldots n$

w users , $w = 1, 2, \ldots n$

a files , $a = 1, 2, \ldots r$

Q_{wa} query traffic between wth user and ath file

U_{wa} update traffic between wth user and ath file

S_{ka} fixed cost of storing and maintaining a copy of the ath file at the kth processor node

C_t admissible link capacities, $t = 0, 1, 2, \ldots k$

$d_{ij}(t)$ cost of a link of the tth capacity and connecting nodes i and j

x_{ij} capacity index assigned to the link between nodes i and j

y_{ka} = 1 if a copy of the ath file is stored at the kth node
= 0 otherwise

$q_{ij}(a,w,k)$ query traffic leaving node i on link (i,j) from wth user and directed to a copy of the ath file located at the kth node.

$u_{ij}(a,w,k)$ update traffic leaving node i on link (i,j) from wth user and directed to a copy of the ath file located at the kth node.

Figure 3 - Casey's second model

problems. For example a 2-file, 18-node problem has 64 billion possible solutions. Casey, therefore, restricts the problem to consider only tree-type networks. Casey attributes the following advantages to tree-type networks: (1) routing decisions are drastically reduced since there is a unique path between any two nodes, (2) a fast iterative algorithm can be developed to calculate the effect of a change in a tree structure, and (3) trees are typically used in many practical situations. Since standard math programming techniques cannot be employed in the solution of this problem, heuristic approaches are developed.

Casey developed an algorithm for finding a "good" netwrok configuration. A brief description of the algorithm is given below. The first step uses the Esau-Williams algorithm to design a minimum cost centralized network having a single copy of each file located at the arbitrarily selected central node. The next step allows duplicate copies of the files and permits distribution of the files in the network. During this step, the topology of the network is held constant. Using the algorithm developed in Casey's first paper, the optimal allocation is found for each file independently. Using this as a starting solution, other allocations are tried by moving the files to neighboring locations. The allocation having the least cost is retained. The third step attempts to redesign the network topology by exchanging links while holding the files in their newly determined locations. The link exchange that results in the greatest cost reduction is put into effect. The second and

third steps are repeated until no further improvement in cost can be obtained.

The design algorithm was tested using a 2-file, 10-node example. It was shown that network cost reductions could be achieved by alternating between the file allocation and the topology design processes. There is, however, a strong correlation between the two processes in that the topology obtained following distribution of the files strongly affects the redistribution of the files.

Chandy and Hewes [5] are concerned with extending previous results of file allocation in a distributed system to include considerations of memory hierarchies at each node in the network. The objective is to develop a model which retains the best features of the Chu [6] and Casey [3] models and, in addition, determines the types of memory devices to purchase for each site in the network in order to minimize overall average response time. The model is developed in three stages where each stage adds additional considerations to the model developed in the previous stage.

The first stage concerns the problem of allocating copies of a single file in a network. This is essentially a reformation of Casey's first model in which the nonlinear objective function is replaced by a linear integer objective function with additional constraints. These constraints guarantee that all queries will be satisfied even if an originating node does not have a copy of the file. In addition, the overall objective is changed from minimizing total cost to minimizing the overall average response

time. In all other respects, the assumptions are the same in both models. The formal mathematical model is given in Figure 4.

Minimize $\sum_{i \neq j} \sum_{j} q_j y_{ij} t_{ij} + \sum_{j} \left(\sum_{i} r_{ij} u_j \right) x_j$

Subject to

$$\sum_{i \neq j} y_{ij} + x_j = 1 \qquad j = 1, \cdots N$$

$$y_{ij} \leq x_i \qquad i \neq j, \; i, j = 1, \cdots N$$

where

q_i	rate at which queries are generated at node j
u_j	rate at which updates are generated at node j
t_{ij}	response time for a query at node j if the query is satisfied by accessing a copy at node i
t_{jj}	= 0 if i = j
r_{ij}	time required to implement an update generated at site i on a copy at site j
X_i	= 1 if a copy of the single file is placed at site i = 0 otherwise
X_{ij}	= 1 if a query at node j is satisfied by accessing a file at node i, i≠j = 0 otherwise
N	number of nodes

Figure 4 - Chandy and Hewes's first model

A branch and bound procedure is used to obtain an optimal solution to the integer programming problem. In order to obtain a good lower bound for this procedure, the integer restriction is relaxed and the corresponding linear programming problem is solved using heuristics to obtain a good starting solution. It is claimed to be more efficient than Casey's algorithm especially

for the solution of very large problems.

In the second stage, the model is extended to include the allocation of multiple files. It appears that the authors were unaware of the work done by Casey in his second paper in which he also considers the same problem. This model incorporates a constraint limiting the total cost of the system by a given bound. The mathematical model is given in Figure 5.

Minimize $$\sum_{i \neq j} \sum_{j} \sum_{k} q_{jk} y_{ijk} t_{ijk} + \sum_{j} \sum_{k} \sum_{\lambda} r_{ijk} u_{jk} x_{jk}$$

Subject to

$$\sum_{i \neq j} \sum_{j} \sum_{k} C_{ijk} y_{ijk} + \sum_{j} \sum_{k} B_{jk} x_{jk} \leq G$$

$$\sum_{i \neq j} y_{ijk} + x_{jk} = 1 \quad \forall\, j,k$$

$$x_{ijk} \leq x_{jk} \qquad \forall\, i,j,k \quad i \neq j$$

where

K number of files in the system

B_{jk} cost of storing a copy of file k in site i

C_{ijk} cost of replying to a query generated at site j regarding file k, if a copy of the file in site i is accessed to satisfy the query

G total budget allowed

x_{jk} = 1 if a copy of file k is stored at site j
= 0 otherwise

y_{ijk} = 1 if a query generated at site j for file k is satisfied by accessing a copy of the file at site i
= 0 otherwise

The variables $q_{jk}, u_{jk}, t_{ijk}, r_{ijk}$ are defined similarly to the variables q_j, u_j, t_{ij}, r_{ij} in the first model particularized to a file k.

Figure 5 - Chandy and Hewes's Second Model

The budget constraint will necessarily bound the minimal access time that can be obtained. The constraint is brought into the objective function using a Lagrange multiplier. This model is solved by first applying the procedure from the first stage to one file at a time. If the total cost obtained in this manner does not satisfy the budget constraint, an iterative procedure is used which varies the Lagrange multiplier and corresponding access times until the budget constraint is satisfied.

The third stage completes the model definition by the addition of considerations of memory hierarchies and their affect on access times. The formal mathematical model is not given (the authors indicate their intention to include it in a leter version),but the solutions are obtained by heuristic methods which find an approximate solution. This solution is obtained by rounding a linear programming solution which allows continuous (rather than discrete) amounts of memory at each site.

Levin and Morgan [9,10] having reviewed the previous work in the modeling of distributed data bases concluded that the problem could be viewed along three dimensions: (1) the level of sharing, (2) the behavior of access patterns, and (3) the level of information on the behavior of access patterns. The level of sharing distinguishes between "data sharing" and "program and data sharing." Data sharing refers to the situation where every node has a copy of each program, but not necessarily the data, that it accesses. Program and data sharing refers to the situation where programs as well as data can be shared among the

nodes. Due to the difficulties involved in maintaining compatible copies of a program at multiple sites, the ability to consider programs as a shared resource may be important. Access pattern behavior may be classified as being either "static" or "dynamic." Static behavior refers to patterns which are fixed over time, whereas dynamic behavior refers to patterns which are changing over time. The level of information available may be "complete" or "partial." When the access request patterns are known with certainity, the information level is said to be complete. When the forecasted patterns may differ from the actual patterns, the information level is said to be partial.

The authors observed that previous models viewed the problem as one of data sharing with static access patterns and complete information. In order to provide more complete coverage of the file allocation problem in the three dimensions, the authors developed four separate models varying the assumptions of access pattern behavior and level of information while including program and data sharing in all. Specifically, the four combinations considered are summarized below:

(1) static, complete

(2) static, partial

(3) dynamic, complete

(4) dynamic partial

In all other respects, these models are very similar to previously developed models. The objective is to minimize overall operating cost as a function or the number of file copies maintained, the location of these copies, and the access request patterns.

As before, the primary focus is on the optimal allocation of data files. Levin and Morgan developed and tested various search procedures and heuristics (along the lines of the previous models) in order to obtain computationally efficient solution methods. Due to the complexity of the models and their similarity to the models previously discussed, the formal mathematical presentations and discussion of detailed solution methods have been omitted.[3]

A unique approach to the distribution of a data base has been presented by Ghosh [8]. Ghosh is interested in distributing a data base with logical associations between distributable segments in order to reduce the response time to satisfy queries. Each segment is assumed to contain all instances of a record type (i.e., segmentation by function). In terms of the three dimensions defined by Levin and Morgan, Ghosh studied data sharing under the assumptions of static access patterns and complete information. This might imply that there was a great deal of similarity between this work and that of Chu, Casey, and Chandy and Hewes. There are, however, several differences concerning the problem to be solved and the approach to be taken.

Ghosh assumes that in order to satisfy queries, access to multiple segments (files) are required. For retrieval purposes, a query can be divided into two parts, the qualification part and the target part, each of which may require access to multiple segments. His paper is directed toward determining an allocation of target segments in order to reduce response time without regard to communications costs through the use of parallel search

---

[3] Interested readers are refered to [9].

techniques. He, therefore, optimizes with respect to the dependencies between segments of a query (segments are accessed together) rather than dependencies between a query and each of its segments as independent entities (segments are accessed separately).

The problem can be summarized as follows: Given a set of queries requiring access to multiple segments, determine an allocation of the segments to the nodes of a computer network such that no two target segments which appear together in a query are resident at the same node. The problem stated in this way only specifies an allocation to satisfy the parallel search requirement. Additional considerations include the issue of redundancy (i.e., do you permit multiple copies of segments), and the determination of the minimal number of nodes which will satisfy the search requirement for a given set of queries.

Ghosh developed several algorithms based upon a set of theorems and the use of combinatorial statistics to determine allocations with and without redundancy. The algorithms find feasible solutions to the parallel search problem. Optimal solutions require the repetitive application of the algorithms to find minimal node solutions since the problem is never formulated as a mathematical model which can be optimized through direct iterative procedures.

We view this approach as being particularly applicable to the distribution of a data base on a computer network where the nodes of the network are all at one site. In this type of environment, communication costs are not a consideration.

We formulate the problem of allocation of segments without redundancy studied by Ghosh as a zero-one integer programming model which determines the minimal number of nodes and the allocation of segments for parallel searching of a given set of queries. The model has three sets of constraints. The first guarantees that each query can be parallel searched, and the second guarantees that exactly one copy of each segment is allocated. The third set of constraints are needed to insure the proper structure for the formulation.

Given:

$Q = \{Q_k \mid k = 1, \cdots M\}$ a set of queries

$T = \{t_i \mid i = 1, \cdots N\}$ a set of target segments

$I_k = \{i \mid t_i \in Q_k\}$ the index set of target segments contained in query k

Variables:

X_{ij} = 1 if t_i is located at node j
= 0 otherwise

Y_j = 1 if node j has at least 1 target segment located there
= 0 otherwise

Minimize:

$$\sum_{j=1}^{N} Y_j$$

Subject to:

$$\sum_{i \in I_k} X_{ij} \leq 1 \quad \forall j, k = 1, \cdots M$$

$$\sum_{j=1}^{N} X_{ij} = 1 \quad i = 1, \cdots N$$

$$\sum_{i=1}^{M} X_{ij} \leq N Y_j \quad j = 1, \cdots N$$

$$X_{ij} = 0, 1 \quad Y_j = 0, 1$$

Conclusions

This paper has presented some considerations in the distribution of a data base and surveyed the research in the application of modeling to the allocation of resources. As discussed in the introduction, allocation of resources cannot occur until they have been partitioned into distributable segments. The partitioning problem is not addressed by any current research. The research (with the exception of Ghosh) views the data base as a collection of independent files. The problems associated with accessing the data base are not considered since application programs are assumed to have direct access to particular files. The files have simple structure (e.g., sequential, indexed sequential) with no complex relationships maintained among them. Ghosh recognized the problem of providing access paths to a data base where complex relationships exist between segments, but he did not consider the problem in his research.

The trend toward data base management systems which are responsible for the creation, accessing, and maintenance of a large collection of information containing complex interrelationships is prevalent in centralized systems today. It seems that the same advantages which led to the growth of data base management systems in centralized systems can be applied in distributed systems as well. Distributed systems, therefore, should not be viewed in terms of data files locked into applications programs but in terms of an integrated database. Most current research (particularly in the area of modeling) does not seem to be addressing the real problems involved in a

distributed system of this type. Research into applying current database technology to distributed systems is just beginning to surface.

A new type of computer configuration for database management systems was proposed by Canady [1] which seems particularly applicable to distributed systems. The basic concept of this configuration is the separation of the database management system and database from the other major software components of a system (e.g., operating system, applications programs). The database management function is implemented on a separate machine which has exclusive access to the database. This machine is referred to as a back-end computer. The other functions of the system reside on a host computer. This configuration offers advantages in the simultaneous sharing of a common database in a computer network especially where the computer systems in the network differ in machine type and configuration. A computer network could consist of back-end machines which serve a number of different hosts. Data can be transferred between very different host machines without reformatting and translation as long as the back-ends are of the same type. In such a network, the previously mentioned problem of partitioning the database among the back-ends would exist. In addition, the relationship between the physical placement of the host machine and the back-end machines in local and/or remote locations would have to be considered. It was shown that a complete database management system (DBMS) based on the DBTG proposal could be implemented in a back-end minicomputer. A choice, therefore, exists on

whether to distribute the DBMS in some type of hierarchy among the back-ends or to install complete copies of the DBMS on each back-end. The experimental implementation of this concept consisted of a single back-end and did not consider the multiple back-end problem.

Peebles and Manning [13] have investigated a computer architecture for distributed data base systems. They were mainly concerned with the design of a message-switched operating system which would support distributed data processing. Their design incorporated the idea of a back-end computer for handling the data base management function. Only the primitive data base architecture of simple direct access file structure has been implemented, but the authors are considering the implementation of a more complex distributed DBMS.

Lowenthal [11] investigated the possibility of implementing a data base management function across a network of minicomputers. He viewed this data management function (DMF) as consisting of a hierarchy of process-oriented modules. He states that the DMF should be designed to treat segmented databases in much the same way as nonsegmented databases. Only the lowest level of the hierarchy should be concerned with the distribution of the data. In this way, a DMF could be upgraded to include support for distributed databases with minimal impact of the rest of the DMF configuration.

Further research into the design of a DBMS which will operate in a distributed environment as current DBMS operate in a

centralized environment is needed. This research must consider the partitioning and distribution of the data resources, the configuration of the system responsible for creating, accessing, and maintaining the data resources, and the determination of access paths which may be complex due to the interdependencies between the distributable segments. The development of models to represent the relationships in a distributed database system which build on the models previously presented would be included in this research. Whether current DBMS can be adapted for supporting distributed systems or whether a DBMS needs to be designed especially for distributed systems needs to be further investigated.

References

1. Canaday, R.H., Harrison, R.D., Ivie, E.L., Ryder, J.L. and Wehr, L.A. "A Back-End Computer for Data Base Management," Communication of the ACM, 17, 10, October 1974, pp. 575-582.

2. Canning, R.G. "In your future: Distributed systems?," EDP Analyzer, Vol. 11, No. 8, August 1973, pp. 1-13.

3. Casey, R.G., "Allocation of copies of a file in an information network," Proceedings of the Spring Joint Computer Conference, 1972, AFIPS Press, Vol. 40, 1972, pp. 617-625.

4. ———— , "Design of tree networks for distributed data," Proceeding of the AFIPS National Computer Conference, 1973, pp. 251-257.

5. Chandy, K.M. and Hewes, J.E. "File Allocation in Distributed Systems," Proceedings International Symposium on Computer Performance Modeling, Measurement, and Evaluation, March, 1976.

6. Chu, W.W. "Optimal File Allocation in a Multiple Computer System," IEEE Transactions on Computers, Vol. C-18, No. 10, October, 1969, pp. 885-889.

7. Chu, W.W. "Optimal File Allocation in a Computer Network," Published in Computer-Communication Network, Prentice Hall, 1973, pp. 82-94.

8. Ghosh, S.P. "Distributing a Data Base with Logical Associations on a Computer Network for Parallel Searching," IBM Report No. RJ 1439 (#22109), IBM Research Laboratory, San Jose, California, August, 1974.

9. Levin, K. "Organizing Distributed Data Bases in Computer Networks," Ph.D. dissertation, Wharton School, University of Pennsylvania, 1974.

10. ——— and Morgan, H.L. "Optimizing distributed data bases- a framework for research, " Proceeding of the Joint Computer Conference, 1975, AFIPS Press, Vol. , pp. 473-478.

11. Lowenthal, E.I., "Computing Subsystems for the Data Management Function", National Computer Conference, May 1974.

12. Merrill, P.K. "Computer Netting and Distributed Data: A Tutorial," Technical Report No. 02.610, IBM System Development Division, San Jose, California, Feb. 1974.

13. Peebles, R. and Manning, E. "A Computer Architecture for Large (Distributed) Data Bases," Department of Computer Science, Computer Communications Networks Group, University of Waterloo.